Implementation of a web application using one of the studied NoSQL databases

Introduction

Goal: The aim is to create a web page that allows the display and ranking of movies from a database, as well as providing functionality to add, edit, and delete a movie.

Overview

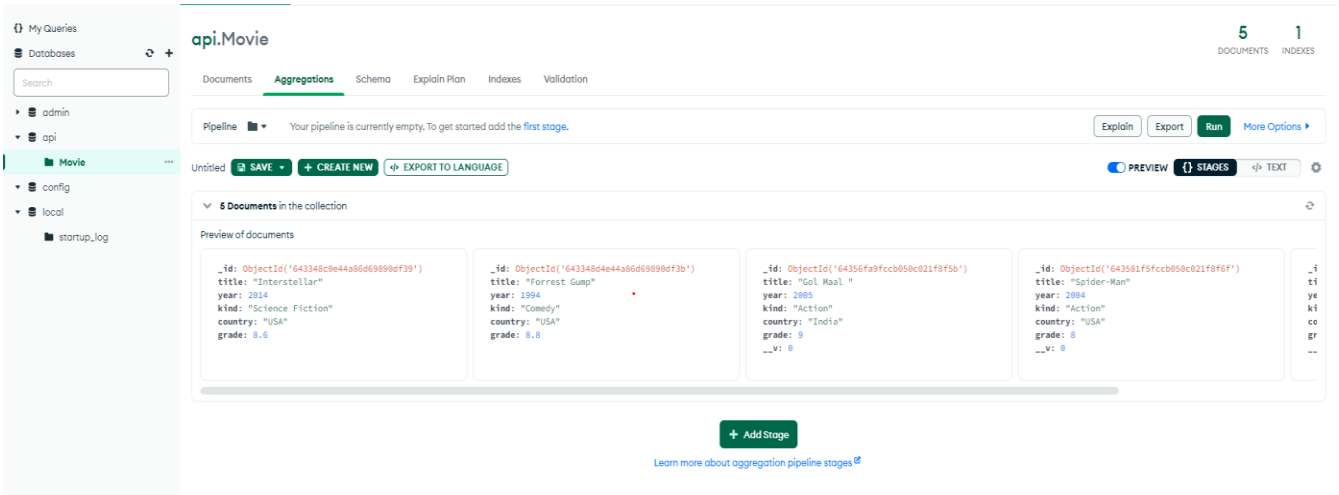
Model for this web page includes a main frame displaying a dropdown list of films sorted in ascending alphabetical order of the title, read-only information about the selected movie, buttons for rating and deleting the displayed movie, and a button to add a new film. The architecture for this project follows the MVC pattern, with the model being the class Film View, a JSP to display the frames, the controller to interpret requests, and a class to connect to the NoSQL database. In this report, we will explore the implementation of this web application using **MongoDB** and provide details on how to test and configure the database.

II. Requirements and Specifications

This project utilizes the full Stack, a comprehensive full stack web development methodology. The combines four essential tools: **MongoDB**, **Express's, AngularJS**, and **NodeJS**. To design the website, we will be following the conventional MVC Architecture, which consists of Model, View, and Controller.

The Model handles the database connection string and database creation, while the View refers to the User Interface, which showcases the website's output. The Controller is responsible for managing all the control data push and fetching processes between the Model and View. Finally, we create the server.js file, which acts as a Rest-API for connecting the User Interface to facilitate data transition.

MongoDB

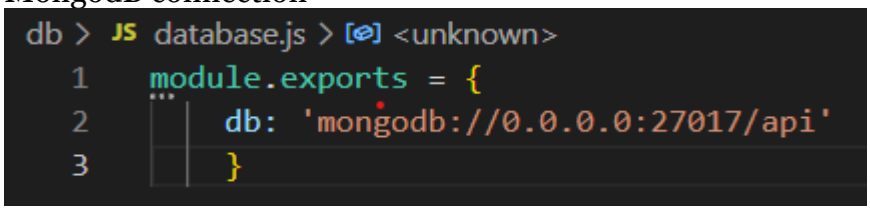


Data in local Host

http://localhost:8080/endpoint


```
[{"_id":"643348c0e44a86d69890df39","title":"Interstellar","year":2014,"kind":"Science Fiction","country":"USA","grade":8.6},{"_id":"643348d4e44a86d69890df3b","title":"Forrest Gump","year":1994,"kind":"Comedy","country":"USA","grade":8.8},
{"_id":"64356fa9fccb050c021f8f5b","title":"Gol Maal ","year":2005,"kind":"Action","country":"India","grade":9,"__v":0},{"_id":"643581f5fccb050c021f8f6f","title":"Spider-Man","year":2004,"kind":"Action","country":"USA","grade":8,"__v":0},
{"_id":"64358272fccb050c021f8f73","title":"The King's Man","year":2021,"kind":"Action","country":"USA","grade":7,"__v":0},{"_id":"643589cffccb050c021f8f91","title":"jam","year":2009,"kind":"action","country":"usa","grade":9,"__v":0}]
```


MongodB connection



IV. Implementation

Develop the front-end UI

With the back-end API in place, we can now develop the front-end UI using AngularJS. This will involve creating views that allow users to view, add, edit, and delete movies from the database.

Started the Local server After installing all required files
http://localhost:4200

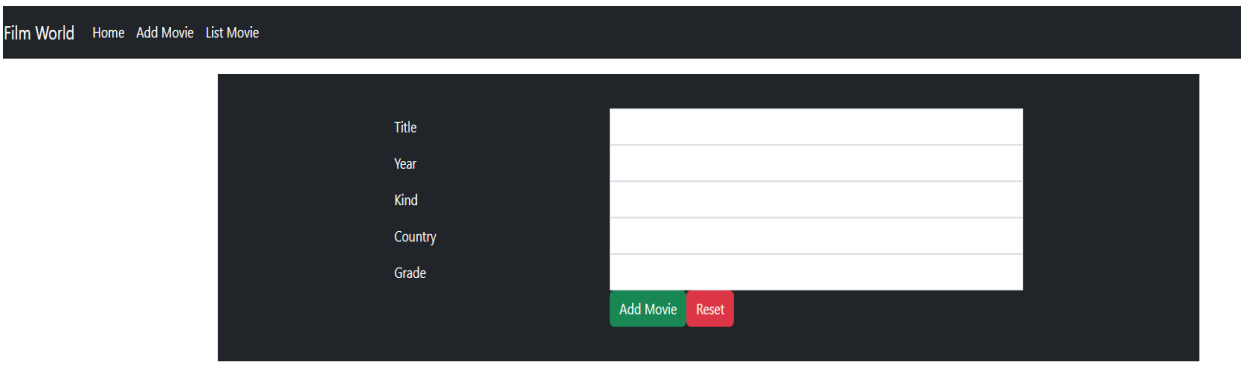
```
C:\FrontEnd\Meanstack>ng serve
√ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   1.71 MB |
polyfills.js          | polyfills     | 314.27 kB |
styles.css, styles.js | styles        | 209.40 kB |
main.js               | main          |  50.93 kB |
runtime.js            | runtime       |   6.52 kB |

                      | Initial Total |   2.28 MB

Build at: 2023-04-11T09:50:00.185Z - Hash: 4e9af387fedd097d - Time: 27117ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
```

Connect the front-end and back-end: The next step is to connect the front-end UI to the back-end API. This will involve making HTTP requests to the API from the UI to retrieve and update movie information in the database.

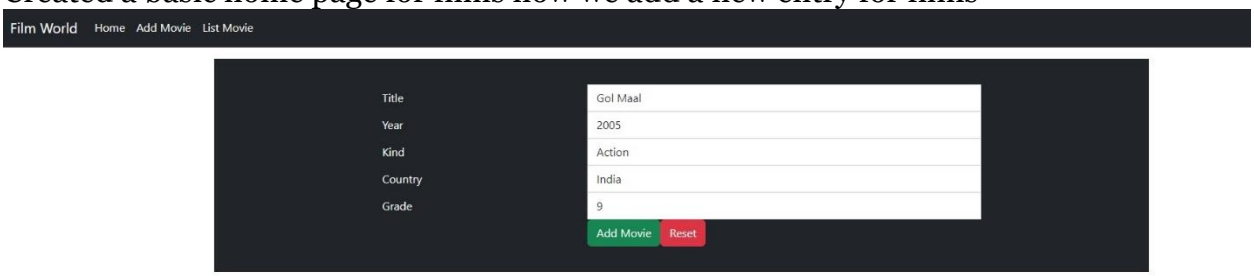
src > app > TS movies.service.ts > ...

```typescript
import { Injectable } from '@angular/core';
import { HttpClient,HttpParams } from '@angular/common/http';
import { Observable } from 'rxjs';
@Injectable({
  providedIn: 'root'
})
export class MoviesService {
  name() {
    throw new Error('Method not implemented.');}
  constructor(private http:HttpClient) { }
  addMovie(movie:any) {
    return this.http.post('http://localhost:8080/endpoint/add-movie', movie);
  }
  listMovie() {
    return this.http.get('http://localhost:8080/endpoint/')
  }
  deleteMovie(id:any)  {
    return this.http.delete('http://localhost:8080/endpoint/del-movie/'+id)
  }
```

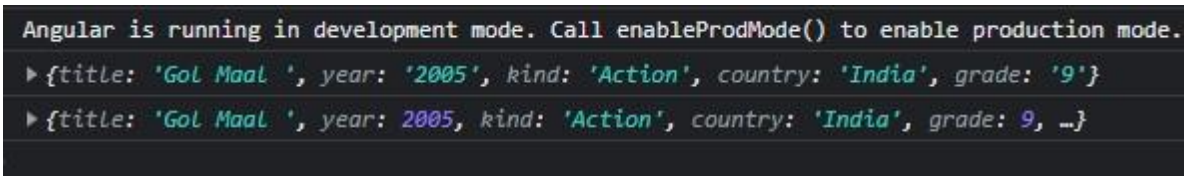
Home page



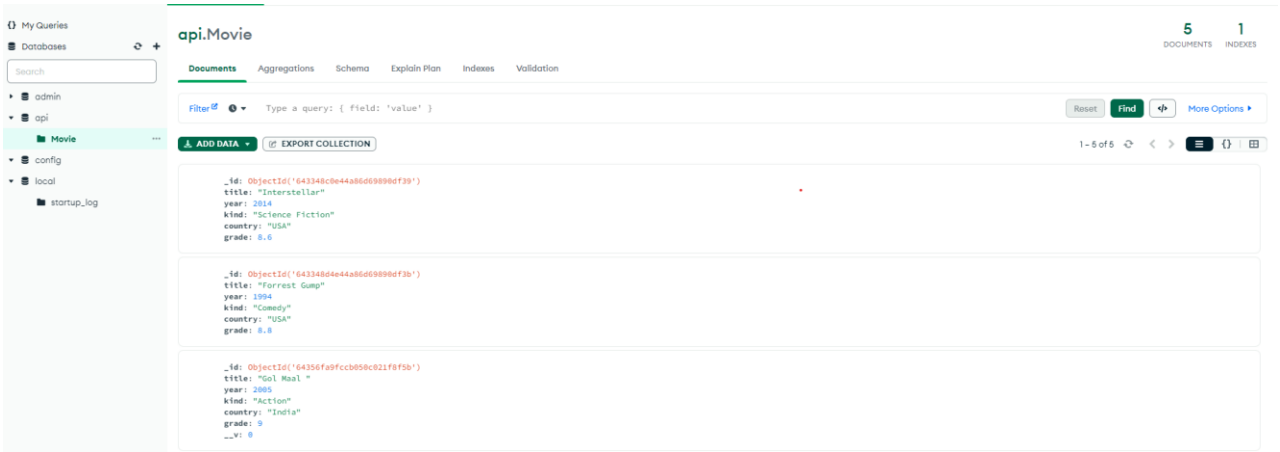
Created a basic home page for films now we add a new entry for films



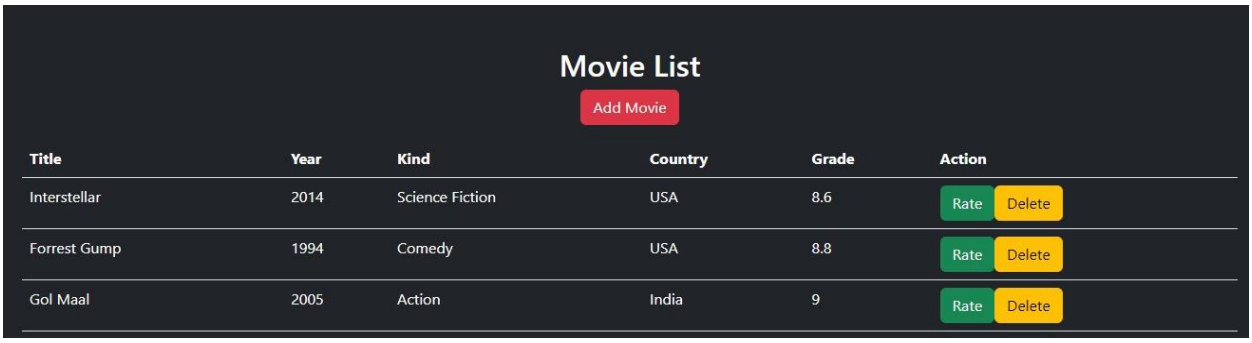
We check in web console of entries



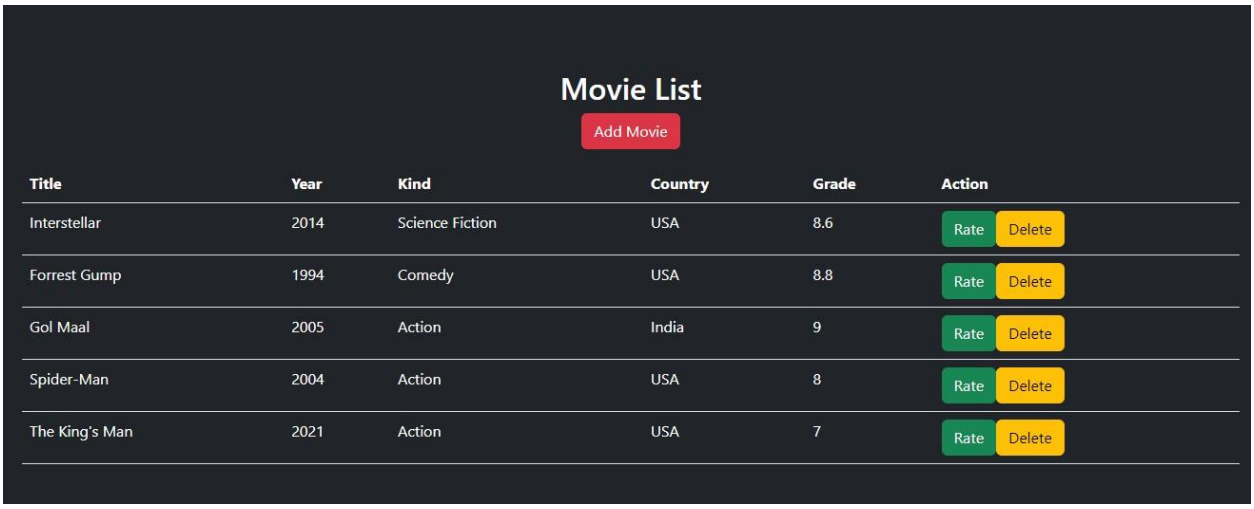
Now we check the in Mondo dB



Now we created a new page Movies after adding the data in front page it directly taking us this page where we have data in one database

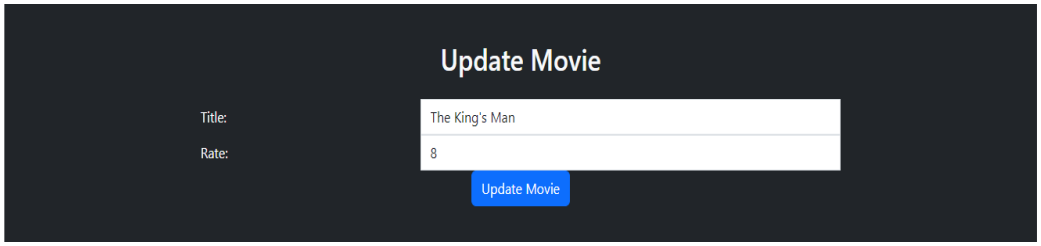


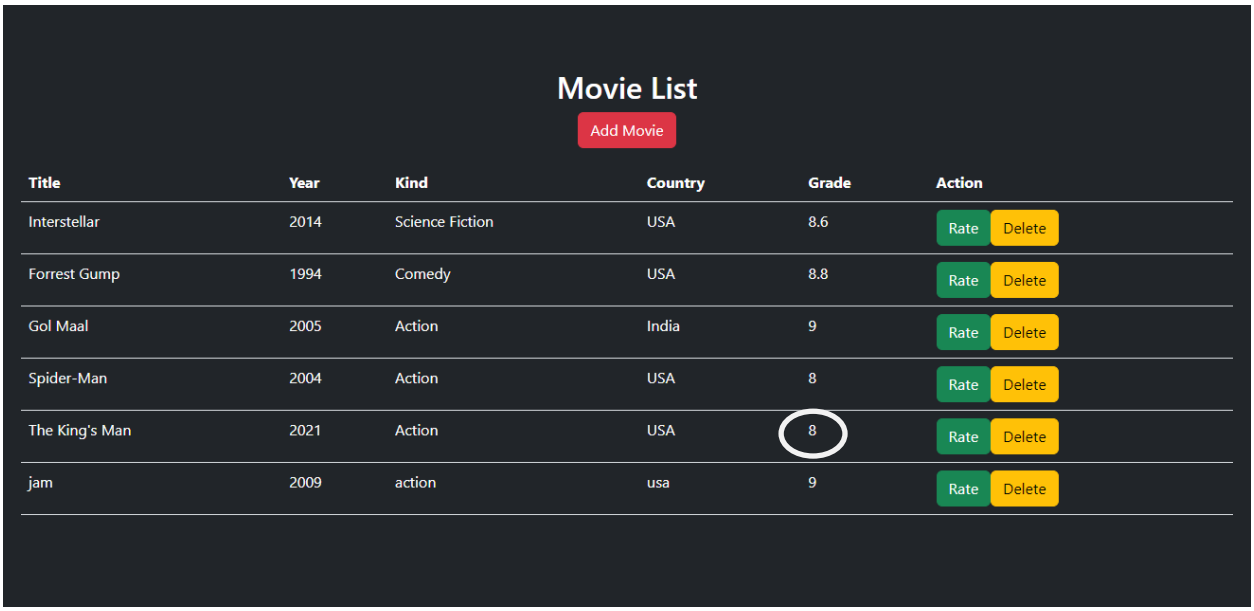
Now to add the two new movies in database in front page and we can see here



We can see two entry of movies spider-Man, The King Man

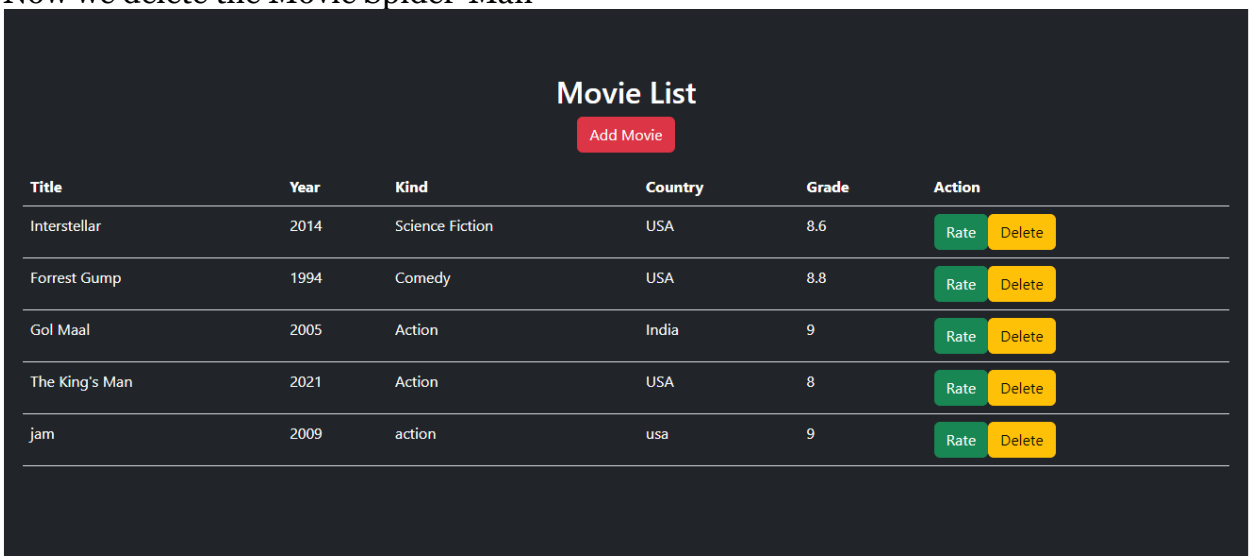
Now to we update the movie rating





Here the Kings Man movie rating is updated

Now we delete the Movie Spider-Man



Conclusion

The above work involves creating a web page for displaying and ranking movies from a database, as well as allowing users to add, edit, and delete movies. The project uses the MEAN stack, which consists of MongoDB, ExpressJS, AngularJS, and NodeJS. The web page is designed using the traditional MVC architecture, with the model handling database connections, the view displaying the user interface, and the controller managing data flow between the model and view.

A server.js file is created to serve as a REST API for connecting the user interface to the data. The implementation process involves setting up the environment, creating the database, designing the user interface, implementing the controller and model.

VIII. References

Project is mainly base on these YouTube video

1.https://www.youtube.com/watch?v=o2HxDnPz7xY&list=PLGt1lxwGVOI4wiOz38XXYWGXbgOkmi_ro&index=7&ab_channel=TutorialRays

2.https://www.youtube.com/watch?v=RU-1JQ2bPFE&list=PLGt1lxwGVOI4wiOz38XXYWGXbgOkmi_ro&index=2&ab_channel=TutorialRays